

Formalizacja naiwnej teorii typów

Agnieszka Kozubek

Instytut Informatyki
Uniwersytet Warszawski

Wrocław, 17 listopada 2011 r.

O czym będzie?

- 1 Motywacja
- 2 Teoria typów
- 3 Mniej naiwna teoria typów
- 4 Typy indukcyjne

Dlaczego uczymy teorii mnogości?

Dlaczego uczymy teorii mnogości?

- nauczyć języka matematyki
- wprowadzić podstawowe pojęcia matematyczne
- nauczyć ścisłego rozumowania
- ćwiczyć abstrakcyjne myślenie

Dlaczego uczymy teorii mnogości?

Dlaczego uczymy teorii mnogości studentów **pierwszego** roku?

- nauczyć języka matematyki
- wprowadzić podstawowe pojęcia matematyczne
- nauczyć ścisłego rozumowania
- ćwiczyć abstrakcyjne myślenie

Dlaczego uczymy teorii mnogości?

Dlaczego uczymy teorii mnogości studentów **pierwszego** roku informatyki?

- nauczyć języka matematyki
- wprowadzić podstawowe pojęcia matematyczne
- nauczyć ścisłego rozumowania
- ćwiczyć abstrakcyjne myślenie

Dlaczego uczymy teorii mnogości?

Dlaczego uczymy teorii mnogości studentów **pierwszego** roku informatyki?

- nauczyć języka matematyki
- wprowadzić podstawowe pojęcia matematyczne
- nauczyć ścisłego rozumowania
- ćwiczyć abstrakcyjne myślenie

Dlaczego uczymy teorii mnogości?

Dlaczego uczymy teorii mnogości studentów **pierwszego** roku informatyki?

- nauczyć języka matematyki
- wprowadzić podstawowe pojęcia matematyczne
- nauczyć ścisłego rozumowania
- ćwiczyć abstrakcyjne myślenie

Dlaczego uczymy teorii mnogości?

Dlaczego uczymy teorii mnogości studentów **pierwszego** roku informatyki?

- nauczyć języka matematyki
- wprowadzić podstawowe pojęcia matematyczne
- nauczyć ścisłego rozumowania
- ćwiczyć abstrakcyjne myślenie

Dlaczego uczymy teorii mnogości?

Dlaczego uczymy teorii mnogości studentów **pierwszego** roku informatyki?

- nauczyć języka matematyki
- wprowadzić podstawowe pojęcia matematyczne
- nauczyć ścisłego rozumowania
- ćwiczyć abstrakcyjne myślenie

Teoria mnogości dziś

- jest językiem współczesnej matematyki
- powszechnie używana jako podstawa matematyki

Nie po to powstała!

Teoria mnogości dziś

- jest językiem współczesnej matematyki
- powszechnie używana jako podstawa matematyki

Nie po to powstała!

Teoria mnogości dziś

- jest językiem współczesnej matematyki
- powszechnie używana jako podstawa matematyki

Nie po to powstała!

Początki teorii mnogości

- paradoksy w naiwnej teorii mnogości
- pytania o niesprzeczność matematyki (początek XX wieku)
- aksjomatyczna teoria mnogości
- powstała jako podstawa matematyki

Początki teorii mnogości

- paradoksy w naiwnej teorii mnogości
- pytania o niesprzeczność matematyki (początek XX wieku)
- aksjomatyczna teoria mnogości
- powstała jako podstawa matematyki

Początki teorii mnogości

- paradoksy w naiwnej teorii mnogości
- pytania o niesprzeczność matematyki (początek XX wieku)
- aksjomatyczna teoria mnogości
- powstała jako podstawa matematyki

Początki teorii mnogości

- paradoksy w naiwnej teorii mnogości
- pytania o niesprzeczność matematyki (początek XX wieku)
- aksjomatyczna teoria mnogości
- powstała jako podstawa matematyki

Cena niesprzeczności

Niskopoziomowa implementacja prostych pojęć

- $7 = 5 \cup \{5, 6\}$
- $\langle a, b \rangle = \{\{a\}, \{a, b\}\}$
- $[a]_r = \{b \mid \langle a, b \rangle \in r\}$

Implementacja przysłania specyfikację

Cena niesprzeczności

Niskopoziomowa implementacja prostych pojęć

- $7 = 5 \cup \{5, 6\}$
- $\langle a, b \rangle = \{\{a\}, \{a, b\}\}$
- $[a]_r = \{b \mid \langle a, b \rangle \in r\}$

Implementacja przysłania specyfikację

Sklejenie

Teoria mnogości skleja dwa pojęcia:

- zbiór jako uniwersum
- zbiór jako materializacja predykatu

Tworzenie dowolnych zbiorów prowadzi do paradoksów.

Aksjomat wycinania

Zamiast $\{x \mid W(x)\}$ mamy $\{x \in A \mid W(x)\}$.

Sklejenie

Teoria mnogości skleja dwa pojęcia:

- zbiór jako uniwersum
- zbiór jako materializacja predykatu

Tworzenie dowolnych zbiorów prowadzi do paradoksów.

Aksjomat wycinania

Zamiast $\{x \mid W(x)\}$ mamy $\{x \in A \mid W(x)\}$.

Czy teoria mnogości to łatwy przedmiot?

WYDAJE SIĘ łatwa.

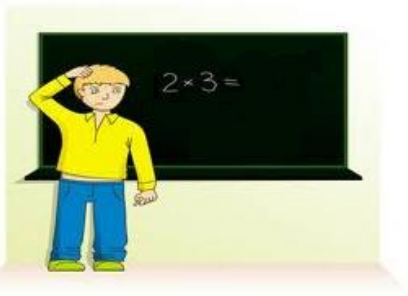
ALE wcale nie jest łatwa do nauczenia się.

Czy teoria mnogości to łatwy przedmiot?

WYDAJE SIĘ łatwa.

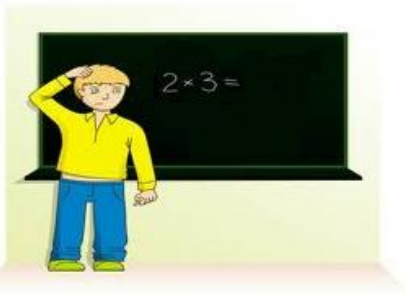
ALE wcale nie jest łatwa do nauczenia się.

Skomplikowany problem



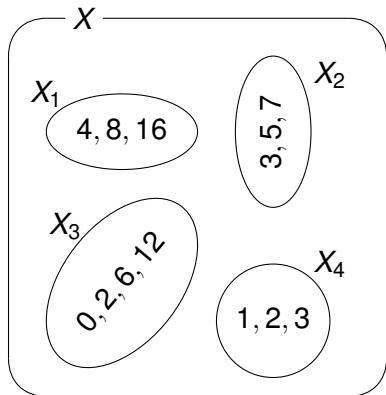
\in czy \subseteq ?

Skomplikowany problem



\in czy \subseteq ?

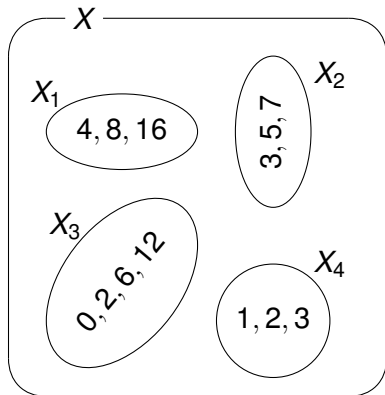
Prosty przykład



$$\begin{array}{ll}
 X \in P(P(\mathbb{N})) & \\
 X_i \in X & X_i \in P(\mathbb{N}) \\
 n \in X_i & n \in \mathbb{N}
 \end{array}$$

W głowie trzeba zbudować system typów.

Prosty przykład



$$X_i \in X$$

$$n \in X_i$$

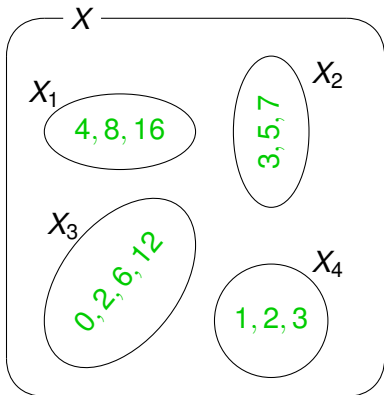
$$X \in P(P(\mathbb{N}))$$

$$X_i \in P(\mathbb{N})$$

$$n \in \mathbb{N}$$

W głowie trzeba zbudować system typów.

Prosty przykład



$$X_i \in X$$

$$n \in X_i$$

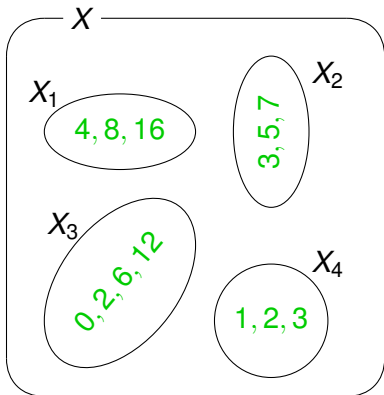
$$X \in P(P(\mathbb{N}))$$

$$X_i \in P(\mathbb{N})$$

$$n \in \mathbb{N}$$

W głowie trzeba zbudować system typów.

Prosty przykład



$$X_i \in X$$

$$n \in X_i$$

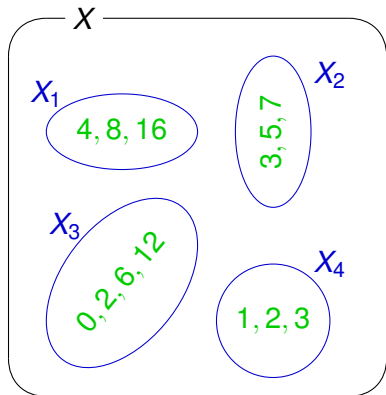
$$X \in P(P(\mathbb{N}))$$

$$X_i \in P(\mathbb{N})$$

$$n \in \mathbb{N}$$

W głowie trzeba zbudować system typów.

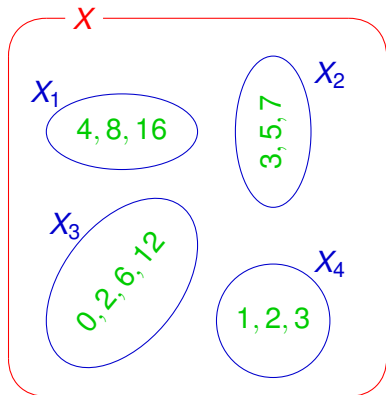
Prosty przykład



$$\begin{array}{ll}
 X_i \in X & X \in P(P(\mathbb{N})) \\
 n \in X_i & X_i \in P(\mathbb{N}) \\
 & n \in \mathbb{N}
 \end{array}$$

W głowie trzeba zbudować system typów.

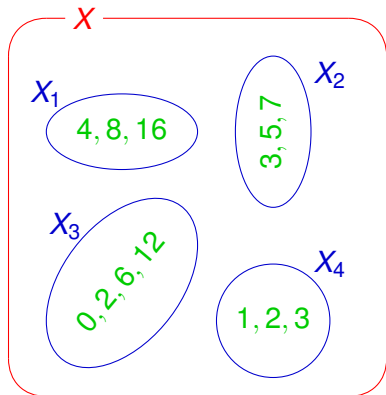
Prosty przykład



$$\begin{array}{ll}
 X_i \in X & X \in P(P(\mathbb{N})) \\
 n \in X_i & X_i \in P(\mathbb{N}) \\
 & n \in \mathbb{N}
 \end{array}$$

W głowie trzeba zbudować system typów.

Prosty przykład



$$\begin{array}{ll}
 X_i \in X & X \in P(P(\mathbb{N})) \\
 n \in X_i & X_i \in P(\mathbb{N}) \\
 & n \in \mathbb{N}
 \end{array}$$

$$A \in \cup X?$$

W głowie trzeba zbudować system typów.

Prosty przykład

$$\begin{array}{ll} X \in P(P(\mathbb{N})) & \\ X_i \in X & X_i \in P(\mathbb{N}) \\ n \in X_i & n \in \mathbb{N} \end{array}$$

$$A \in UX?$$

W głowie trzeba zbudować system typów.

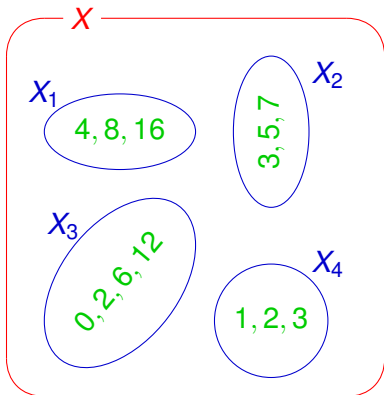
Prosty przykład

$$\begin{array}{ll} X \in P(P(\mathbb{N})) & \\ X_i \in X & X_i \in P(\mathbb{N}) \\ n \in X_i & n \in \mathbb{N} \end{array}$$

$$A \in \cup X?$$

W głowie trzeba zbudować system typów.

Prosty przykład

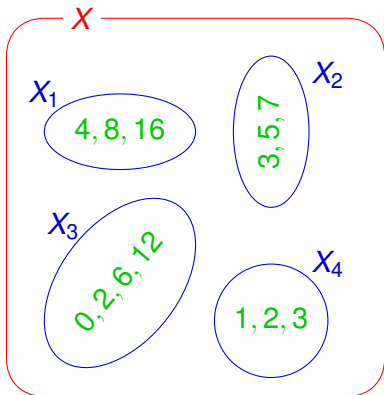


$$\begin{array}{ll}
 X_i \in X & X \in P(P(\mathbb{N})) \\
 n \in X_i & X_i \in P(\mathbb{N}) \\
 & n \in \mathbb{N}
 \end{array}$$

$$A \in \cup X?$$

W głowie trzeba zbudować system typów.

Prosty przykład

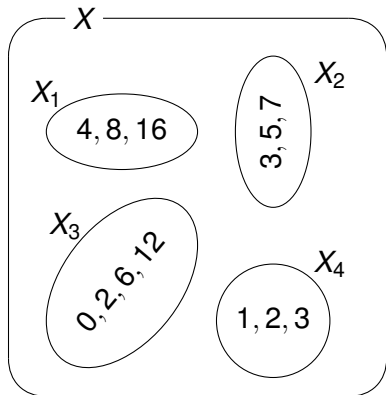


$$\begin{array}{ll}
 X_i \in X & X \in P(P(\mathbb{N})) \\
 n \in X_i & X_i \in P(\mathbb{N}) \\
 & n \in \mathbb{N}
 \end{array}$$

$$A \in \cup X?$$

W głowie trzeba zbudować system typów.

Prosty przykład



$$\begin{array}{ll}
 X_i \in X & X \in P(P(\mathbb{N})) \\
 n \in X_i & X_i \in P(\mathbb{N}) \\
 & n \in \mathbb{N}
 \end{array}$$

$$A \in \cup X?$$

W głowie trzeba zbudować **system typów**.

Co chcę zrobić?

Cel

Stworzyć system typów, który sformalizuje naiwną teorię typów.

Co chcę zrobić?

- rozwinąć teorię, w której można uprawiać matematykę (elementarną)
- użyteczna w nauczaniu podstaw matematyki
- rozróżnia uniwersum od predykatu
- bez sztucznych kodowań
- intensjonalne specyfikacje zamiast dowodliwych własności
- oparta na teorii typów

Co chcę zrobić?

- rozwinąć teorię, w której można uprawiać matematykę (elementarną)
- użyteczna w nauczaniu podstaw matematyki
- rozróżnia uniwersum od predykatu
- bez sztucznych kodowań
- intensjonalne specyfikacje zamiast dowodliwych własności
- oparta na teorii typów

Co chcę zrobić?

- rozwinąć teorię, w której można uprawiać matematykę (elementarną)
- użyteczna w nauczaniu podstaw matematyki
- rozróżnia uniwersum od predykatu
- bez sztucznych kodowań
- intensjonalne specyfikacje zamiast dowodliwych własności
- oparta na teorii typów

Co chcę zrobić?

- rozwinąć teorię, w której można uprawiać matematykę (elementarną)
- użyteczna w nauczaniu podstaw matematyki
- rozróżnia uniwersum od predykatu
- bez sztucznych kodowań
- intensjonalne specyfikacje zamiast dowodliwych własności
- oparta na teorii typów

Co chcę zrobić?

- rozwinąć teorię, w której można uprawiać matematykę (elementarną)
- użyteczna w nauczaniu podstaw matematyki
- rozróżnia uniwersum od predykatu
- bez sztucznych kodowań
- intensjonalne specyfikacje zamiast dowodliwych własności
- oparta na teorii typów

Co chcę zrobić?

- rozwinąć teorię, w której można uprawiać matematykę (elementarną)
- użyteczna w nauczaniu podstaw matematyki
- rozróżnia uniwersum od predykatu
- bez sztucznych kodowań
- intensjonalne specyfikacje zamiast dowodliwych własności
- oparta na teorii typów

Rachunek lambda

- początki w latach 30-tych XX wieku (Church, Curry)
- alternatywna podstawa matematyki
- model obliczeń
- podstawowe pojęcie: funkcja
- funkcja rozumiana w sposób intensjonalny

Składnia rachunku lambda

Lambda term

- zmienne x, y, \dots są lambda termami
- jeśli M, N są lambda termami, to MN także jest lambda termem
- jeśli x jest zmienną, M jest lambda termem, to $\lambda x.M$ jest lambda termem

Znaczenie

Każdy lambda term reprezentuje funkcję.

- zmienne x, y, \dots reprezentują jakieś funkcje
- MN jest aplikacją funkcji M do argumentu N
- $\lambda x.M$ to funkcja z parametrem x i ciałem M

Składnia rachunku lambda

Lambda term

- zmienne x, y, \dots są lambda termami
- jeśli M, N są lambda termami, to MN także jest lambda termem
- jeśli x jest zmienną, M jest lambda termem, to $\lambda x.M$ jest lambda termem

Znaczenie

Każdy lambda term reprezentuje funkcję.

- zmienne x, y, \dots reprezentują jakieś funkcje
- MN jest aplikacją funkcji M do argumentu N
- $\lambda x.M$ to funkcja z parametrem x i ciałem M

Beta redukcja

$$f : \mathbb{N} \rightarrow \mathbb{N}$$

$$f = \lambda x : \mathbb{N}. x + 7$$

$$f(5) = (\lambda x : \mathbb{N}. x + 7)(5) \rightarrow 5 + 7 \rightarrow 12$$

redukcja term w postaci normalnej
term który się nie redukuje

Beta redukcja

$$f : \mathbb{N} \rightarrow \mathbb{N}$$

$$f = \lambda x : \mathbb{N}. x + 7$$

$$f(5) = (\lambda x : \mathbb{N}. x + 7)(5) \rightarrow 5 + 7 \rightarrow 12$$

redukcja term w postaci normalnej
term który się nie redukuje

Beta redukcja

$$f : \mathbb{N} \rightarrow \mathbb{N}$$

$$f = \lambda x : \mathbb{N}. x + 7$$

$$f(5) = (\lambda x : \mathbb{N}. x + 7)(5) \rightarrow 5 + 7 \rightarrow 12$$

redukcja

term w postaci normalnej
term który
się nie redukuje

Beta redukcja

$$f : \mathbb{N} \rightarrow \mathbb{N}$$

$$f = \lambda x : \mathbb{N}. x + 7$$

$$f(5) = (\lambda x : \mathbb{N}. x + 7)(5) \rightarrow 5 + 7 \rightarrow 12$$

redukcja

term w postaci normalnej
term który
się nie redukuje

Beta redukcja

$$f : \mathbb{N} \rightarrow \mathbb{N}$$

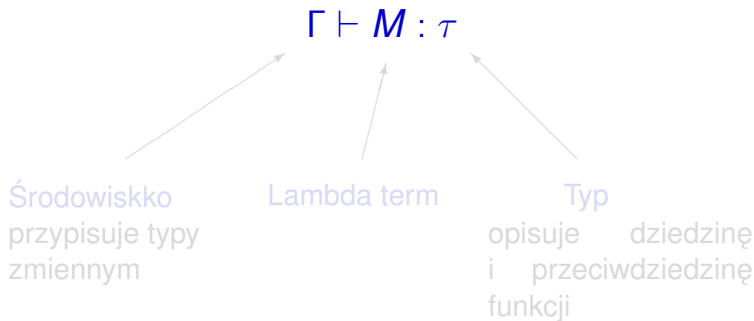
$$f = \lambda x : \mathbb{N}. x + 7$$

$$f(5) = (\lambda x : \mathbb{N}. x + 7)(5) \rightarrow 5 + 7 \rightarrow 12$$

redukcja

term w postaci normalnej
term który
się nie redukuje

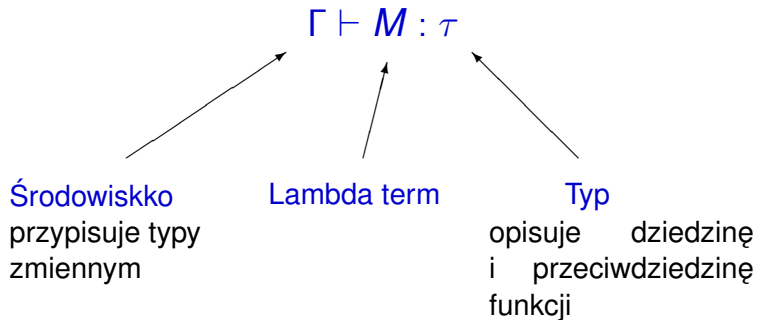
Systemy przypisania typów



Reguły typowania

Reguły mówiące, jak przypisać typy lambda termom.

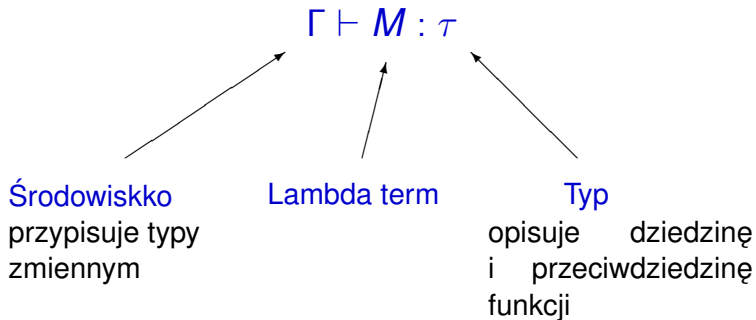
Systemy przypisania typów



Reguły typowania

Reguły mówiące, jak przypisać typy lambda termom.

Systemy przypisania typów



Reguły typowania

Reguły mówiące, jak przypisać typy lambda termom.

Problem inhabitacji

Problem inhabitacji (niepustości typu)

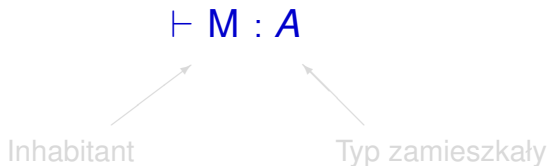
Czy istnieje lambda term M taki, że $\vdash M : A$?

$\vdash ? : A$

Problem inhabitacji

Problem inhabitacji (niepustości typu)

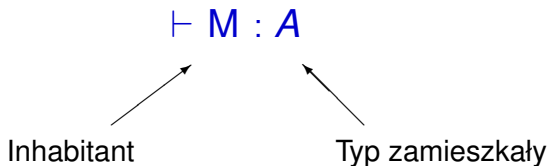
Czy istnieje lambda term M taki, że $\vdash M : A$?



Problem inhabitacji

Problem inhabitacji (niepustości typu)

Czy istnieje lambda term M taki, że $\vdash M : A$?



Podstawowy system typów

Rachunek lambda z typami prostymi λ_{\rightarrow}

$$\Gamma, x : A \vdash x : A$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \rightarrow B}$$

$$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

Reguła aplikacji

funkcja
z A do B

argument
typu A

$$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

aplikacja funkcji

Reguła aplikacji

Reguła aplikacji

funkcja

z A do B

argument

typu A

$$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

aplikacja funkcji

Reguła aplikacji

Reguła aplikacji

funkcja

z A do B

argument

typu A

$$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

aplikacja funkcji

Reguła aplikacji

Reguła aplikacji

funkcja

z A do B

argument

typu A

$$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

aplikacja funkcji

Reguła aplikacji

Reguła aplikacji

funkcja
z A do B

argument
typu A

$$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

aplikacja funkcji

Reguła aplikacji

Reguła aplikacji

funkcja
z A do B

argument
typu A

$$\frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

aplikacja funkcji

Reguła aplikacji

Reguła aplikacji

$$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

Wygląda znajomo?

Reguła aplikacji

$$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

Reguła modus ponens

Reguła aplikacji

dowód $A \rightarrow B$

dowód A

$$\frac{\Gamma \vdash p : A \rightarrow B \quad \Gamma \vdash q : A}{\Gamma \vdash p(q) : B}$$

dowód B

Reguła modus ponens

Eureka!

Izomorfizm Curry'ego-Howarda!

Izomorfizm Curry'ego - Howarda

Rachunek lambda

rachunek lambda
z typami prostymi λ_{\rightarrow}

typy
reguły typowania
inhabitacja

$\vdash ? : \tau$

typy zamieszkałe

Logika

minimalna zdaniowa
logika (intuicjonistyczna)

formuły
reguły dowodzenia
dowodliwość

$\vdash \tau ?$

formuły dowodliwe

Izomorfizm Curry'ego - Howarda

Rachunek lambda

rachunek lambda
z typami prostymi $\lambda \rightarrow$

typy

reguły typowania

inhabitacja

$\vdash ? : \tau$

typy zamieszkałe

Logika

minimalna zdaniowa
logika (intuicjonistyczna)

formuły

reguły dowodzenia

dowodliwość

$\vdash \tau ?$

formuły dowodliwe

Izomorfizm Curry'ego - Howarda

Rachunek lambda

rachunek lambda
z typami prostymi λ_{\rightarrow}

typy

reguły typowania
inhabitacja

$\vdash ? : \tau$

typy zamieszkałe

Logika

minimalna zdaniowa
logika (intuicjonistyczna)

formuły

reguły dowodzenia
dowodliwość

$\vdash \tau ?$

formuły dowodliwe

Izomorfizm Curry'ego - Howarda

Rachunek lambda

rachunek lambda
z typami prostymi λ_{\rightarrow}

typy

reguły typowania

inhabitacja

$\vdash ? : \tau$

typy zamieszkałe

Logika

minimalna zdaniowa
logika (intuicjonistyczna)

formuły

reguły dowodzenia

dowodliwość

$\vdash \tau ?$

formuły dowodliwe

Izomorfizm Curry'ego - Howarda

Rachunek lambda

rachunek lambda
z typami prostymi λ_{\rightarrow}

typy
reguły typowania

inhabitacja

$\vdash ? : \tau$

typy zamieszkałe

Logika

minimalna zdaniowa
logika (intuicjonistyczna)

formuły
reguły dowodzenia

dowodliwość

$\vdash \tau ?$

formuły dowodliwe

Izomorfizm Curry'ego - Howarda

Rachunek lambda

rachunek lambda
z typami prostymi λ_{\rightarrow}

typy
reguły typowania
inhabitacja

$\vdash ? : \tau$

typy zamieszkałe

Logika

minimalna zdaniowa
logika (intuicjonistyczna)

formuły
reguły dowodzenia
dowodliwość

$\vdash \tau ?$

formuły dowodliwe

Redukcja dla dowodów

*Lemma*₁ : $A \rightarrow B$, *Lemma*₂ : A

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1 (lem_2))$ *Lemma*₁ *Lemma*₂ : B

Przebiegamy od dowodu $A \rightarrow B$ i dowodu A do dowodu B .
 Aby udowodnić B , stosujemy *Lemma*₁ do *Lemma*₂.
 Dowód B uzyskujemy z *Lemma*₁ i dowodu A .
*Lemma*₁ jest dowodem $A \rightarrow B$. Zatem udowodnimy B .

*Lemma*₁ (*Lemma*₂) : B

Aby udowodnić B , stosujemy *Lemma*₁ do *Lemma*₂.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B$, $Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1 (lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy wziąć lem_1 , zastosować go do lem_2 i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem udowodniliśmy B .

$Lemma_1 (Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B$, $Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1 (lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy wziąć lem_1 , zastosować go do lem_2 i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem udowodniliśmy B .

$Lemma_1 (Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B$, $Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1 (lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy wziąć lem_1 , zastosować go do lem_2 i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem udowodniliśmy B .

$Lemma_1 (Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B$, $Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1 (lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy **wziąć lem_1** , zastosować go do lem_2 i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem udowodniliśmy B .

$Lemma_1 (Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B$, $Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1(lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy wziąć lem_1 , **zastosować go do lem_2** i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem udowodniliśmy B .

$Lemma_1(Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B, Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1 (lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy wziąć lem_1 , zastosować go do lem_2 i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem udowodniliśmy B .

$Lemma_1 (Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B$, $Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1 (lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy wziąć lem_1 , zastosować go do lem_2 i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem udowodniliśmy B .

$Lemma_1 (Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B$, $Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1 (lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy wziąć lem_1 , zastosować go do lem_2 i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem udowodniliśmy B .

$Lemma_1 (Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B$, $Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1 (lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy wziąć lem_1 , zastosować go do lem_2 i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem **udowodniliśmy B** .

$Lemma_1 (Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

Redukcja dla dowodów

$Lemma_1 : A \rightarrow B, Lemma_2 : A$

$(\lambda lem_1 : A \rightarrow B \lambda lem_2 : A. lem_1(lem_2)) Lemma_1 Lemma_2 : B$

Przypuśćmy, że lem_1 dowodzi $A \rightarrow B$ a lem_2 dowodzi A . Wtedy możemy wziąć lem_1 , zastosować go do lem_2 i otrzymać dowód B . Zauważmy, że $Lemma_1$ jest dowodem $A \rightarrow B$ a $Lemma_2$ jest dowodem A . Zatem udowodniliśmy B .



$Lemma_1(Lemma_2) : B$

Aby udowodnić B , stosujemy $Lemma_1$ do $Lemma_2$.

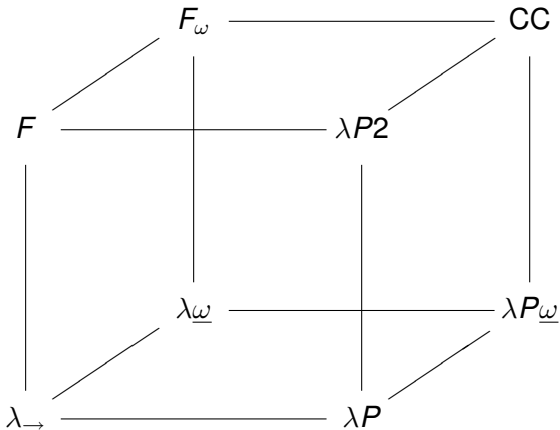
Systemy dowodzenia twierdzeń

- Coq, Agda, Isabelle, HOL, Matita, Epigram ...
- oparte na teorii typów i izomorfizmie Curry'ego-Howarda

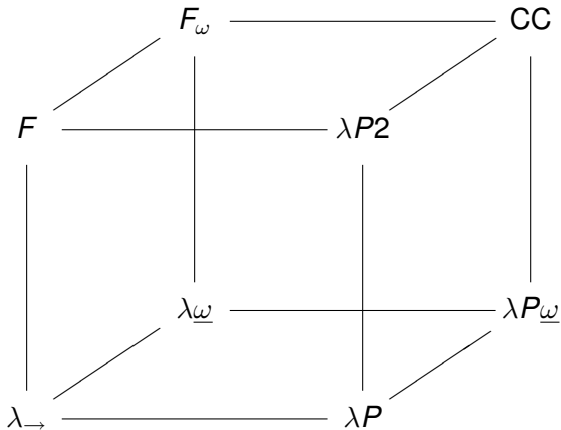
Rozszerzenia systemu z typami prostymi

- system λP – logika pierwszego rzędu
- system F – logika zdaniowa drugiego rzędu
- system F_ω – logika zdaniowa wyższego rzędu
- rachunek konstrukcji CC – rachunek predykatów wyższego rzędu

Kostka Barendregta



Kostka Barendregta



Pure
Type
Systems

Pure Type Systems (PTSs)

- baza dla teorii typów
- współczesny formalizm do definiowania teorii typów
- ułatwia porównywanie systemów typów
- parametryczny
- opisuje jakie konstrukcje są dopuszczalne
- także systemy spoza kostki Barendregta
- używany do definiowania nowych systemów typów (nowych logik)

Reguły typowania dla PTS

$$\text{(Var)} \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \quad \text{(Weak)} \frac{\Gamma \vdash A : B \quad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}$$

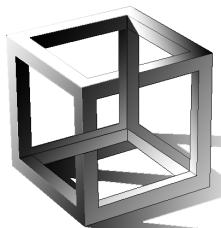
$$\text{(App)} \frac{\Gamma \vdash F : (\Pi x : A. B) \quad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x := a]}$$

$$\text{(Abs)} \frac{\Gamma, x : A \vdash b : B \quad \Gamma \vdash (\Pi x : A. B) : s}{\Gamma \vdash \lambda x : A. b : (\Pi x : A. B)}$$

$$\text{(Prod)} \frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A. B : s_3} \quad (s_1, s_2, s_3) \in R$$

$$\text{(Conv)} \frac{\Gamma \vdash A : B \quad \Gamma \vdash B' : s \quad B =_{\beta} B'}{\Gamma \vdash A : B'}$$

Podstawowe zagrożenie

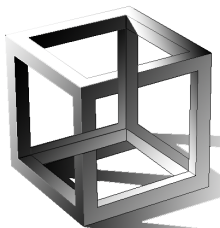


Czy system jest niesprzeczny?

Paradoksy

- paradoks Russella,
- paradoks Girarda.

Podstawowe zagrożenie

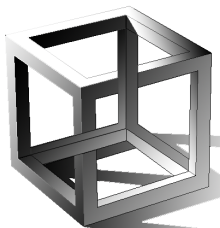


Czy system jest niesprzeczny?

Paradoksy

- paradoks Russella,
- paradoks Girarda.

Podstawowe zagrożenie

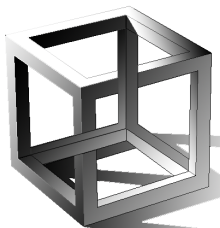


Czy system jest niesprzeczny?

Paradoksy

- paradoks Russella,
- paradoks Girarda.

Podstawowe zagrożenie



Czy system jest niesprzeczny?

Paradoksy

- paradoks Russella,
- paradoks Girarda.

Naiwna teoria typów

System NTT

$$\mathcal{S} = *, \square$$

$$\mathcal{A} = * : \square$$

$$\mathcal{R} = (*, *), (*, \square), (*, \square, *)$$

Sprzeczny system typów

Każdy typ jest zamieszkały \Rightarrow wszystko można udowodnić.

Naiwna teoria typów

System NTT

$$\mathcal{S} = *, \square$$

$$\mathcal{A} = * : \square$$

$$\mathcal{R} = (*, *), (*, \square), (*, \square, *)$$

Sprzeczny system typów

Każdy typ jest zamieszkały \Rightarrow wszystko można udowodnić.

Mniej naiwna teoria typów (LNTT)

- poprawiona wersja naiwnej teorii typów
- moralnie równoważna naiwnej teorii typów
- rozróżnienie na świat logiki i świat obiektów
- podobieństwo do systemu Coq (Prop i Set)

Mniej naiwna teoria typów (LNTT)

- poprawiona wersja naiwnej teorii typów
- moralnie równoważna naiwnej teorii typów
- rozróżnienie na świat logiki i świat obiektów
- podobieństwo do systemu Coq (Prop i Set)

Mniej naiwna teoria typów (LNTT)

- poprawiona wersja naiwnej teorii typów
- moralnie równoważna naiwnej teorii typów
- rozróżnienie na świat logiki i świat obiektów
- podobieństwo do systemu Coq (Prop i Set)

Mniej naiwna teoria typów (LNTT)

- poprawiona wersja naiwnej teorii typów
- moralnie równoważna naiwnej teorii typów
- rozróżnienie na świat logiki i świat obiektów
- podobieństwo do systemu Coq (Prop i Set)

System LNTT

System LNTT

$$\mathcal{S} = *^t, *^p, \Box^t, \Box^p$$

$$\mathcal{A} = *^t : \Box^t, *^p : \Box^p$$

$$\mathcal{R} = (*^t, *^t), (*^p, *^p), (*^t, *^p), (*^t, \Box^t), (\Box^p, *^p), (*^t, \Box^p, *^t)$$

przestrzeń funkcyjna, implikacja, logika predykatów, typy zależne, logika wyższego rzędu, podzbiory jako obiekty

System LNTT

System LNTT

$$\mathcal{S} = *^t, *^p, \Box^t, \Box^p$$

$$\mathcal{A} = *^t : \Box^t, *^p : \Box^p$$

$$\mathcal{R} = (*^t, *^t), (*^p, *^p), (*^t, *^p), (*^t, \Box^t), (\Box^p, *^p), (*^t, \Box^p, *^t)$$

przestrzeń funkcyjna, implikacja, logika predykatów, typy zależne, logika wyższego rzędu, podzbiory jako obiekty

System LNTT

System LNTT

$$\mathcal{S} = *^t, *^p, \Box^t, \Box^p$$

$$\mathcal{A} = *^t : \Box^t, *^p : \Box^p$$

$$\mathcal{R} = (*^t, *^t), (*^p, *^p), (*^t, *^p), (*^t, \Box^t), (\Box^p, *^p), (*^t, \Box^p, *^t)$$

przestrzeń funkcyjna, implikacja, logika predykatów, typy zależne, logika wyższego rzędu, podzbiory jako obiekty

System LNTT

System LNTT

$$\mathcal{S} = *^t, *^p, \Box^t, \Box^p$$

$$\mathcal{A} = *^t : \Box^t, *^p : \Box^p$$

$$\mathcal{R} = (*^t, *^t), (*^p, *^p), (*^t, *^p), (*^t, \Box^t), (\Box^p, *^p), (*^t, \Box^p, *^t)$$

przestrzeń funkcyjna, implikacja, logika predykatów, typy zależne, logika wyższego rzędu, podzbiory jako obiekty

System LNTT

System LNTT

$$\mathcal{S} = *^t, *^p, \square^t, \square^p$$

$$\mathcal{A} = *^t : \square^t, *^p : \square^p$$

$$\mathcal{R} = (*^t, *^t), (*^p, *^p), (*^t, *^p), (*^t, \square^t), (\square^p, *^p), (*^t, \square^p, *^t)$$

przestrzeń funkcyjna, implikacja, logika predykatów, typy zależne, logika wyższego rzędu, podzbiory jako obiekty

System LNTT

System LNTT

$$\mathcal{S} = *^t, *^p, \square^t, \square^p$$

$$\mathcal{A} = *^t : \square^t, *^p : \square^p$$

$$\mathcal{R} = (*^t, *^t), (*^p, *^p), (*^t, *^p), (*^t, \square^t), (\square^p, *^p), (*^t, \square^p, *^t)$$

przestrzeń funkcyjna, implikacja, logika predykatów, typy zależne, logika wyższego rzędu, podzbiory jako obiekty

System LNTT

System LNTT

$$\mathcal{S} = *^t, *^p, \square^t, \square^p$$

$$\mathcal{A} = *^t : \square^t, *^p : \square^p$$

$$\mathcal{R} = (*^t, *^t), (*^p, *^p), (*^t, *^p), (*^t, \square^t), (\square^p, *^p), (*^t, \square^p, *^t)$$

przestrzeń funkcyjna, implikacja, logika predykatów, typy zależne, logika wyższego rzędu, podzbiory jako obiekty

Pytanie

Jak można pokazać, że system jest niesprzeczny?

Silna normalizacja

Twierdzenie

Żaden dowód fałszu nie ma postaci normalnej.

Wniosek

Aby udowodnić niesprzeczność systemu typów, wystarczy udowodnić, że nie ma w nim termów, które nie mają postaci normalnej.

Własność silnej normalizacji (Terminacja)

Każdy ciąg redukcji jest skończony = Każdy term na postać normalną.

Silna normalizacja

Twierdzenie

Żaden dowód fałszu nie ma postaci normalnej.

Wniosek

Aby udowodnić niesprzeczność systemu typów, wystarczy udowodnić, że nie ma w nim termów, które nie mają postaci normalnej.

Własność silnej normalizacji (Terminacja)

Każdy ciąg redukcji jest skończony = Każdy term na postać normalną.

Silna normalizacja

Twierdzenie

Żaden dowód fałszu nie ma postaci normalnej.

Wniosek

Aby udowodnić niesprzeczność systemu typów, wystarczy udowodnić, że nie ma w nim termów, które nie mają postaci normalnej.

Własność silnej normalizacji (Terminacja)

Każdy ciąg redukcji jest skończony = Każdy term na postać normalną.

Translacja do rachunku konstrukcji

Twierdzenie (Coquand, 1988)

CC ma własność silnej normalizacji.

$$\begin{array}{ccc} \text{LNTT} & \xrightarrow{T} & \text{CC} \\ \Gamma \vdash M : A & & T(\Gamma) \vdash T(M) : T(A) \\ M \rightarrow_{\beta} N & & T(M) \rightarrow_{\beta}^+ T(N) \end{array}$$

Tak naprawdę translacje są dwie.

Translacja do rachunku konstrukcji

Twierdzenie (Coquand, 1988)

CC ma własność silnej normalizacji.

LNTT \xrightarrow{T} CC

$\Gamma \vdash M : A$

$T(\Gamma) \vdash T(M) : T(A)$

$M \rightarrow_{\beta} N$

$T(M) \rightarrow_{\beta}^{+} T(N)$

Tak naprawdę translacje są dwie.

Translacja do rachunku konstrukcji

Twierdzenie (Coquand, 1988)

CC ma własność silnej normalizacji.

$$\begin{array}{ccc} \text{LNTT} & \xrightarrow{T} & \text{CC} \\ \Gamma \vdash M : A & & T(\Gamma) \vdash T(M) : T(A) \\ M \rightarrow_{\beta} N & & T(M) \rightarrow_{\beta}^{+} T(N) \end{array}$$

Tak naprawdę translacje są dwie.

Translacja do rachunku konstrukcji

Twierdzenie (Coquand, 1988)

CC ma własność silnej normalizacji.

$$\begin{array}{ccc} \text{LNTT} & \xrightarrow{T} & \text{CC} \\ \Gamma \vdash M : A & & T(\Gamma) \vdash T(M) : T(A) \\ M \rightarrow_{\beta} N & & T(M) \rightarrow_{\beta}^+ T(N) \end{array}$$

Tak naprawdę translacje są dwie.

Translacja do rachunku konstrukcji

Twierdzenie (Coquand, 1988)

CC ma własność silnej normalizacji.

$$\begin{array}{ccc} \text{LNTT} & \xrightarrow{T} & \text{CC} \\ \Gamma \vdash M : A & & T(\Gamma) \vdash T(M) : T(A) \\ M \rightarrow_{\beta} N & & T(M) \rightarrow_{\beta}^+ T(N) \end{array}$$

Tak naprawdę translacje są dwie.

Wniosek

Twierdzenie

LNTT jest niesprzeczna.

Dowód

Silna normalizacja przez translację do rachunku konstrukcji.

Wniosek

Twierdzenie

LNTT jest niesprzeczna.

Dowód

Silna normalizacja przez translację do rachunku konstrukcji.

Siła wyrazu LNTT

Logika

\perp , $A \rightarrow B$, $A \wedge B$, $A \vee B$, $\forall x.\varphi$, $\exists y.\varphi$, $\forall P.\varphi(P)$

Potrzebne są typy danych.

Typy indukcyjne

- pozwala dodać liczby naturalne, listy, wartości logiczne, ...
- podobny do rachunku konstrukcji indukcyjnych (Coq)

Składnia typów indukcyjnych

$$\text{Ind}(X : A)\{\vec{C}\} \mid \text{Constr}(n, l)\vec{N} \mid \text{Elim}(l, Q, M)\{\vec{f}\}$$

Typy indukcyjne

$$\text{Nat} = \text{Ind}(X : *^t)\{X \mid X \rightarrow X\}$$

Obiekty indukcyjne

$$0 = \text{Constr}(0, \text{Nat})$$

$$1 = \text{Constr}(1, \text{Nat})\text{Constr}(0, \text{Nat})$$

Eliminacja obiektów indukcyjnych

$$\text{Elim}(\text{Nat}, \text{Nat}, m)\{0 \Rightarrow 0 \mid Sn \Rightarrow n\}$$

Składnia typów indukcyjnych

$$\text{Ind}(X : A)\{\vec{C}\} \mid \text{Constr}(n, l)\vec{N} \mid \text{Elim}(l, Q, M)\{\vec{f}\}$$

Typy indukcyjne

$$\text{Nat} = \text{Ind}(X : *^t)\{X \mid X \rightarrow X\}$$

Obiekty indukcyjne

$$0 = \text{Constr}(0, \text{Nat})$$

$$1 = \text{Constr}(1, \text{Nat})\text{Constr}(0, \text{Nat})$$

Eliminacja obiektów indukcyjnych

$$\text{Elim}(\text{Nat}, \text{Nat}, m)\{0 \Rightarrow 0 \mid Sn \Rightarrow n\}$$

Składnia typów indukcyjnych

$$\text{Ind}(X : A)\{\vec{C}\} \mid \text{Constr}(n, l)\vec{N} \mid \text{Elim}(l, Q, M)\{\vec{f}\}$$

Typy indukcyjne

$$\text{Nat} = \text{Ind}(X : *^t)\{X \mid X \rightarrow X\}$$

Obiekty indukcyjne

$$0 = \text{Constr}(0, \text{Nat})$$

$$1 = \text{Constr}(1, \text{Nat})\text{Constr}(0, \text{Nat})$$

Eliminacja obiektów indukcyjnych

$$\text{Elim}(\text{Nat}, \text{Nat}, m)\{0 \Rightarrow 0 \mid Sn \Rightarrow n\}$$

Składnia typów indukcyjnych

$$\text{Ind}(X : A)\{\vec{C}\} \mid \text{Constr}(n, l)\vec{N} \mid \text{Elim}(l, Q, M)\{\vec{f}\}$$

Typy indukcyjne

$$\text{Nat} = \text{Ind}(X : *^t)\{X \mid X \rightarrow X\}$$

Obiekty indukcyjne

$$0 = \text{Constr}(0, \text{Nat})$$

$$1 = \text{Constr}(1, \text{Nat})\text{Constr}(0, \text{Nat})$$

Eliminacja obiektów indukcyjnych

$$\text{Elim}(\text{Nat}, \text{Nat}, m)\{0 \Rightarrow 0 \mid Sn \Rightarrow n\}$$

Nowe reguły

Iota redukcja

$$\text{Elim}(\text{Nat}, \text{Nat}, 0)\{f_0 \mid f_1\} \rightarrow_{\iota} f_0$$

$$\text{Elim}(\text{Nat}, \text{Nat}, Sn)\{f_0 \mid f_1\} \rightarrow_{\iota} f_1 \ n \ \text{Elim}(\text{Nat}, \text{Nat}, n)\{f_0 \mid f_1\}$$

Reguły typowania

...

...

...

Nowe reguły

Iota redukcja

$$\text{Elim}(\text{Nat}, \text{Nat}, 0)\{f_0 \mid f_1\} \rightarrow_{\iota} f_0$$

$$\text{Elim}(\text{Nat}, \text{Nat}, Sn)\{f_0 \mid f_1\} \rightarrow_{\iota} f_1 \ n \ \text{Elim}(\text{Nat}, \text{Nat}, n)\{f_0 \mid f_1\}$$

Reguły typowania

...

...

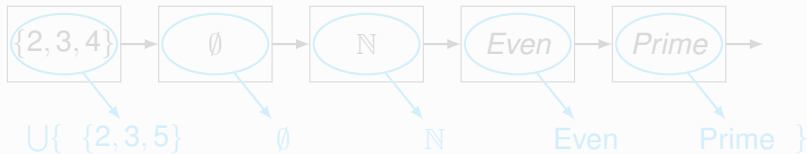
...

Nowe paradoksy

Warunek pozytywności

Typy konstruktorów muszą spełniać (syntaktyczny) warunek pozytywności.

Silna eliminacja na dużych konstruktorach

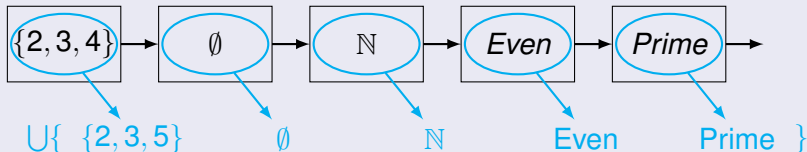


Nowe paradoksy

Warunek pozytywności

Typy konstruktorów muszą spełniać (syntaktyczny) warunek pozytywności.

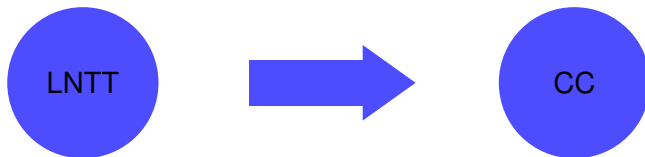
Silna eliminacja na dużych konstruktorach



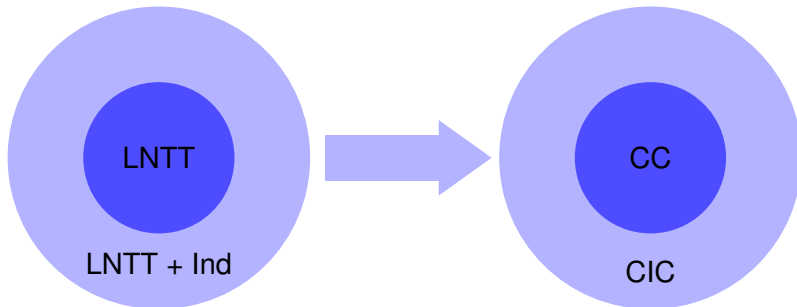
Pytanie

Czy LNTT z typami indukcyjnymi jest niesprzeczny?

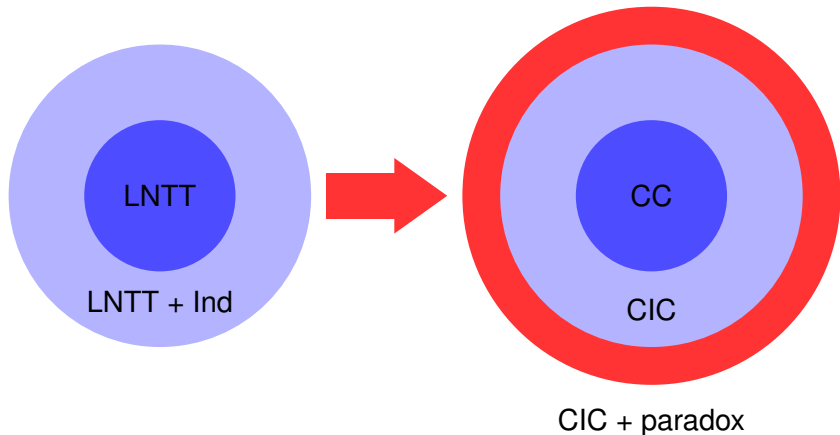
Translacja nie działa



Translacja nie działa



Translacja nie działa



Główny wynik

Twierdzenie

LNTT z typami indukcyjnymi ma własność silnej normalizacji

Dowód

Metoda kandydatów Girarda.

Dziękuję za uwagę.